A unified model with inertia shaping for highly dynamic jumps of legged robots

A Unified Model with Inertia Shaping for Highly Dynamic Jumps of Legged Robots

Ke Wang[1], Guiyang Xin[2], Songyan Xin[2], Michael Mistry[2], Sethu Vijayakumar[2] and Petar Kormushev[1]

***Abstract*— To achieve highly dynamic jumps of legged robots, it is essential to control the rotational dynamics of the robot. In this paper, we aim to improve the jumping performance by proposing a unified model for planning highly dynamic jumps that can approximately model the centroidal inertia. This model abstracts the robot as a single rigid body for the base and point masses for the legs. The model is called the Lump Leg Single Rigid Body Model (LL-SRBM) and can be used to plan motions for both bipedal and quadrupedal robots. By taking the effects of leg dynamics into account, LL-SRBM provides a computationally efficient way for the motion planner to change the centroidal inertia of the robot with various leg configurations. Concurrently, we propose a novel contact detection method by using the norm of the average spatial velocity. After the contact is detected, the controller is switched to force control to achieve a soft landing. Twisting jump and forward jump experiments on the bipedal robot SLIDER and quadrupedal robot ANYmal demonstrate the improved jump performance by actively changing the centroidal inertia. These experiments also show the generalization and the robustness of the integrated planning and control framework.**

I. INTRODUCTION

Early studies on highly dynamic motion such as jumping and running in legged robots are largely influenced by the heuristic control implemented on Raibert's hoppers [1]. The hopper is able to achieve the dynamic behaviors through simple composition of a set of simple controllers that control hopping height, speed, and posture separately. This is possible due to their prismatic leg design which differs from most humanoid robots with human-like morphology. Despite its success, heuristic control is quite limited since it needs large amount of work on parameter tuning. More recently, model based approaches are becoming more and more popular. The spring loaded inverted pendulum (SLIP) model is a well recognized template model for running and jump. An approximated SLIP model has been used to represent the translational motion, robust running and jump are planned with model predictive control [2]. 3D-SLIP model has been used to generate high speed running motion [3] and long jump [4]. However, the SLIP model only considers the point mass dynamics and the angular momentum is ignored. It is therefore difficult to consider motions involving body rotation, such as a twisting jump.

Compared to humanoid robots, quadrupedal robots are often built with lower degree of freedom (DoF) legs and point feet. This equips them with lightweight legs which are often ignored during the motion planning phase or the control phase without introducing significant modeling error. With the assumption of massless legs, dynamic gaits including trotting, pronking, bounding and pacing have been demonstrated on Mini Cheetah through online convex model predictive control [5], [6], [7]. The convex formulation is made possible due to a small angle assumption for roll and pitch angle of the Single Rigid Body Model (SRBM) and predefined footstep locations. Unfortunately, these assumptions do not hold for generating versatile and highly dynamic motions. Recently, a more versatile and acrobatic motion generation framework, the kino-dynamic planner, is proposed in [8]. However, the kino-dynamic planner is complex and computationally inefficient. Therefore, a good trade-off between model complexity and computation efficiency is needed.

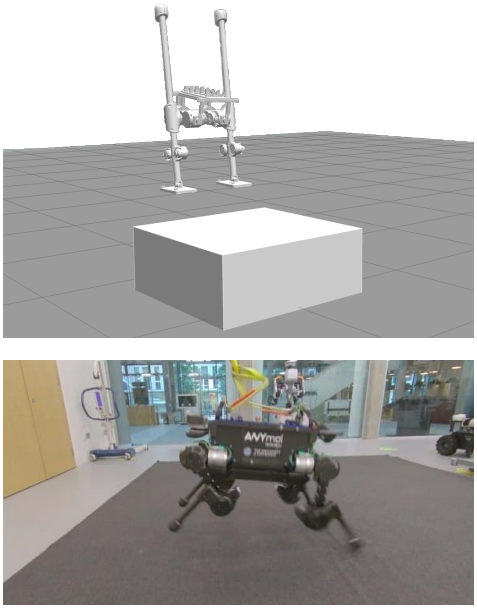


Fig. 1: Highly dynamic jump motions can be generated for both bipedal and quadrupedal robots using the LL-SRBM. Top: SLIDER robot. Bottom: ANYmal robot.

At the same time, the robustness of the controller is important for guaranteeing the successful execution of the dynamic motion. The robot experiences a large impact when landing; these large contact forces and fast changing velocities can easily make the robot unstable. Roboticists have made various efforts to reduce the effect of the impact. One direction is changing the reference trajectory in real time. [9] shows a quadrupedal robot walking with 50 Hz re-

[1]Robot Intelligence Lab, Dyson School of Design Engineering, Imperial College London, UK. Email: k.wang17@imperial.ac.uk
[2]School of Informatics, University of Edinburgh, Edinburgh, UK.

planning with a slightly simplified SRBM. But no results have been shown with more dynamic motions such as jumping. In [10], [11] the impacts are explicitly modelled, however the simplified model and large uncertainties of state estimation make the control more difficult. [12] presents a feedback controller in an impact invariant space. Although this approach is not affected by the rapid change of velocities, it can not reduce the effects caused by the impact. To reduce the effect of the impact, good contact detection with a soft landing is a promising direction. External force sensors are commonly used to monitor contacts for robot arms [13]. But in practice force signals are too noisy for legged robots. The average spatial velocity, first introduced by [14], is a synthetic representation of all link velocities and an indicator for the kinetic energy of the robot. We will rely on the average spatial velocity for contact detection.

A. Contributions

In this paper, we aim to improve the performance of highly dynamic jumps with the LL-SRBM and the robustness of the tracking controller with contact detection. Additionally, we show that the proposed planning and control methods can be applied to both bipedal and quadrupedal robots. The twisting jump and forward jump are demonstrated in simulation and real robot experiments. Here, we would like to highlight the contributions of the paper:

- A general framework that can generate highly dynamic jumping motions for both bipedal and quadrupedal robots. Each layer of the framework is generalized for both types of robots, including the lump leg concept.
- The Lumped Leg Single Rigid Body Model (LL-SRBM) provides a computationally efficient way to optimize the shape of the robot's inertia while planning the jump motion. By taking the motion of legs into consideration, the jump motion generated by the LL-SRBM achieves better performance (e.g. faster twisting jump, more stable forward jump etc.) than the motion generated by using the SRBM.
- Safe landing is ensured by using a novel contact detection method and switching to force control in landing phase. We propose to use the norm of the average spatial velocity to detect contacts. By taking account of all link states, the average spatial velocity is more robust to noises and general enough for detecting impulses from all directions. After the contact is detected, contact force tracking control is switched on for a period to achieve soft landing. We cannot use force control for the whole jump process since the LL-SRBM is still an approximated model for planning.

II. SYSTEM OVERVIEW

The highly dynamic jumping behaviour is achieved via a hierarchical structure which includes the motion planner and the whole body controller as shown in Fig. 2. The high-level user selects the motion (e.g. twisting jump, forward jump, etc.) and inputs to the motion planner with initial and final states and contact positions for non-flight phases. The

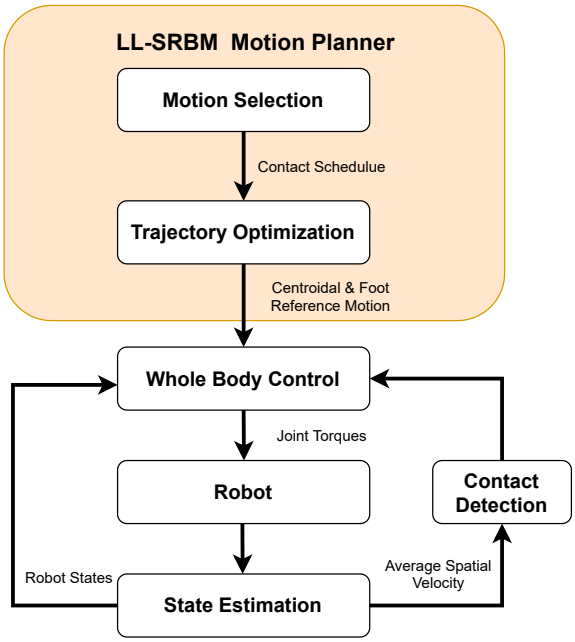


Fig. 2: The planning and control framework.

trajectory optimization with LL-SRBM outputs the reference centroidal and foot trajectories to the whole body controller. The whole body controller computes optimal joint torques for the robot to track these task space targets while considering all physical constraints. The contact detection runs after the robot takes off and monitors the average spatial velocity. Once the contact is detected, the whole body controller switches to force control for a short time to achieve a soft landing. After landing, trajectory tracking is reinstated to achieve the desired steady finishing pose.

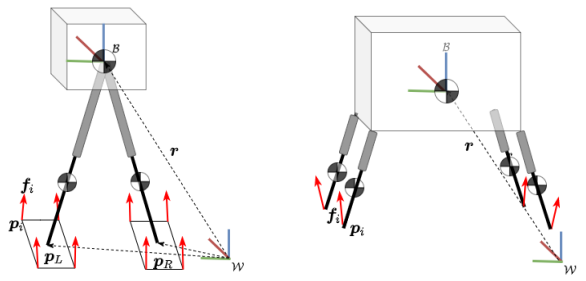
III. THE LUMP LEG SINGLE RIGID BODY MODEL

We are interested in generating high fidelity dynamic jump motions while keeping the model simple enough. The novel aspect of LL-SRBM model is that it abstracts the robot as a single rigid body for the base and point masses for the legs, as illustrated in Fig. 3. The point mass of the leg is located at the CoM of the leg at the default configuration and the distance between the point mass and the leg contact location is proportional to the leg length. Compared to SRBM, the LL-SRBM models the leg dynamics as well, which is important for computing the centroidal inertia, as show in Fig. 4. The full Newton-Euler dynamics of the LL-SRBM with multiple contacts with the environment can be written as:

$$\begin{bmatrix} \dot{\boldsymbol{H}} = \sum_{i=1}^{n} \boldsymbol{f}_i + m\boldsymbol{g} \\ \dot{\boldsymbol{L}} = \sum_{i=1}^{n} (\boldsymbol{p}_i - \boldsymbol{r}) \times \boldsymbol{f}_i \end{bmatrix} \tag{1}$$

where $\boldsymbol{H}$ and $\boldsymbol{L}$ are the linear and angular momentum of the model, $\boldsymbol{f}_i$ is the ith ground reaction force acting on $\boldsymbol{p}_i$, n forces are assumed. $\boldsymbol{r}$ is the position of the CoM, m and $\boldsymbol{g}$ are the mass of the model and the gravity vector. The linear momentum is directly related to the translational velocity of the CoM:

$$\dot{\boldsymbol{r}} = \boldsymbol{H}/m \tag{2}$$



(a) (b)

Fig. 3: The LL-SRBM is a general model for bipedal and quadrupedal robots, it takes the leg dynamics into consideration. It also unifies planar, line and point feet with the distributed Ground Reaction Force representation. (a) shows LL-SRBM applied to a bipedal robot with planar feet. The ground reaction forces are located at four corners of the foot with arbitrary directions. (b) shows LL-SRBM applied to a quadrupedal robot with point feet. For both figures $\mathcal{W}$ and $\mathcal{B}$ stand for the world frame and the body frame.

For LL-SRBM, the CoM is computed as

$$\boldsymbol{r} = \frac{\boldsymbol{r}_B \cdot m_B + \sum_i \boldsymbol{r}_l^i \cdot m_l^i}{m_B + \sum_i m_l^i} \tag{3}$$

where $\boldsymbol{r}_B$ and m_B are CoM and mass of the body, $\boldsymbol{r}_l^i$ and m_l^i are the position and mass of the ith leg.

Angular momentum of the LL-SRBM, $\boldsymbol{L}$, can be expressed as:

$$\boldsymbol{L} = \boldsymbol{I}_W \boldsymbol{\omega} \tag{4}$$

where $\boldsymbol{I}_W$ represents the centroidal inertia of the model in the world frame and $\boldsymbol{\omega}$ is the angular velocity of the body. The centroidal inertia matrix, also called the centroidal composite rigid body inertia (CCRBI) matrix in [14] can be computed as:

$$\boldsymbol{I}_W = \boldsymbol{R}_B \boldsymbol{I}_G \boldsymbol{R}_B^T = \boldsymbol{R}(\boldsymbol{q}_B) \boldsymbol{I}_G \boldsymbol{R}^T(\boldsymbol{q}_B) \tag{5}$$

where $\boldsymbol{R}(\boldsymbol{q}_B)$ is a transformation from local to world frame as given in the Appendix. $\boldsymbol{I}_G \in \mathbb{R}^{6\times6}$ is the centroidal inertia matrix at CoM frame and can be computed as:

$$\boldsymbol{I}_G = \boldsymbol{I}_B + \sum_i \boldsymbol{I}_l^i = \boldsymbol{I}_B + \sum_i m_i (\boldsymbol{r} - \boldsymbol{p}_i + \Delta \boldsymbol{p}_i)^2 \tag{6}$$

where $\boldsymbol{I}_B$ is the inertia of the single rigid body projected to the CoM, and $\Delta \boldsymbol{p}_i$ is the offset between the leg's contact location and the position of leg mass, the second term represents the inertia of legs projected to CoM.

In [5] the Euler angles are used to represent the orientation, however this approach suffers from the problem of gimbal lock, so we prefer quaternion representation for the orientation. The dynamics of the quaternion $\boldsymbol{q}_B$ can be expressed as:

$$\dot{\boldsymbol{q}}_B = \frac{1}{2} \boldsymbol{q}_B \circ \boldsymbol{\omega} = \boldsymbol{Q}(\boldsymbol{q}_B) \boldsymbol{\omega} \tag{7}$$

where $\circ$ represents the quaternion product and $\boldsymbol{Q}(\boldsymbol{q}_B) \in \mathbb{R}^{4\times3}$ is the corresponding matrix representation. By associating equation (4), (5), (7), we can get the dynamics of orientation represented by quaternion:

$$\dot{\boldsymbol{q}}_B = \frac{1}{2} \boldsymbol{Q}(\boldsymbol{q}_B) \boldsymbol{I}_W^{-1}(\boldsymbol{q}_B) \boldsymbol{L} \tag{8}$$

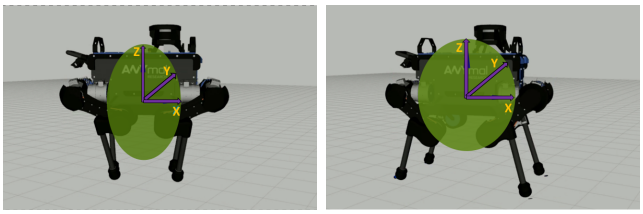
So the complete dynamics of the LL-SRBM can be summarized as:

$$\begin{bmatrix} \dot{\boldsymbol{r}} = \boldsymbol{H}/m \\ \dot{\boldsymbol{q}}_B = \frac{1}{2} \boldsymbol{Q}(\boldsymbol{q}_B) \boldsymbol{I}_W^{-1}(\boldsymbol{q}_B) \boldsymbol{L} \\ \dot{\boldsymbol{H}} = \sum_{i=1}^n \boldsymbol{f}_i + m\boldsymbol{g} \\ \dot{\boldsymbol{L}} = \sum_{i=1}^n (\boldsymbol{p}_i - r) \times \boldsymbol{f}_i \end{bmatrix} \tag{9}$$

with

$$\boldsymbol{r} = \frac{\boldsymbol{r}_B \cdot m_B + \sum_i \boldsymbol{r}_l^i \cdot m_l^i}{m_B + \sum_i m_l^i} \tag{3}$$

$$\boldsymbol{I}_W = \boldsymbol{R}(\boldsymbol{q}_B)(\boldsymbol{I}_B + \sum_i m_i (\boldsymbol{r} - \boldsymbol{p}_i + \Delta \boldsymbol{p}_i)^2) \boldsymbol{R}^T(\boldsymbol{q}_B) \tag{10}$$



(a) (b)

Fig. 4: The shape of the centroidal inertia of the robot is affected by the leg configuration, which can be modelled by LL-SRBM. (a) and (b) show two different leg configurations when ANYmal is performing a twisting jump. Compared with the centroidal inertia in (b), the centroidal inertia in (a) has smaller values on y and z axes.

IV. TRAJECTORY OPTIMIZATION

This section describes the mathematical formulation of trajectory optimization and how we formulate trajectory optimization into a Nonlinear Programming (NLP) problem.

A. Problem Formulation

The trajectory optimization [15] is a powerful tool to find optimal trajectories for complex tasks that contain nonlinear dynamics and involves multiple phases. In this paper, we employ the *multiple shooting method* to transcribe our trajectory optimization problem into a NLP formulation. In the formulation, multiple phases are considered since we are focusing on versatile dynamic jumps that contains multiple pre-defined contact phases: *takeoff*, *flight phase* and *post-landing phase*.

The system has been discretised into N segments and therefore $N+1$ knots exists. Since m phases are assumed, each phase gets N/m segments. Because timing is important for generating highly dynamic motions, we do not fix the time for each phase ΔT. However we keep the same time interval dt between knots inside each phase to reduce the computation cost. For knot k, the state $\boldsymbol{x}[k]$ and control $\boldsymbol{u}[k]$ are defined as:

$$\begin{aligned} \boldsymbol{x}[k] &= [\boldsymbol{r}[k], \boldsymbol{q}_B[k], \boldsymbol{H}[k], \boldsymbol{L}[k]] \\ \boldsymbol{u}[k] &= [\boldsymbol{f}_1[k], ..., \boldsymbol{f}_{N_l}[k], \boldsymbol{p}_1[k], ..., \boldsymbol{p}_{N_l}[k]] \\ \boldsymbol{\xi}[k] &= [\boldsymbol{x}[k], \boldsymbol{u}[k], dt[k]] \end{aligned}$$

where $dt[k]$ is the time interval from knot k to $k+1$, N_l is the leg number, $\boldsymbol{f}_1[k], ..., \boldsymbol{f}_{N_l}[k]$ are the ground reaction forces acting on the feet. For bipedal robots with planar feet, the left foot and right foot collect four corresponding corner forces: $\boldsymbol{f}_L[k] = [\boldsymbol{f}_{L_1}[k], \boldsymbol{f}_{L_2}[k], \boldsymbol{f}_{L_3}[k], \boldsymbol{f}_{L_4}[k]]$, $\boldsymbol{f}_R[k] = [\boldsymbol{f}_{R_1}[k], \boldsymbol{f}_{R_2}[k], \boldsymbol{f}_{R_3}[k], \boldsymbol{f}_{R_4}[k]]$ as shown in Fig. 3a. $\boldsymbol{p}_L[k]$ and $\boldsymbol{p}_R[k]$ are the center position of the left foot and right foot respectively. For quadrupedal robots, $\boldsymbol{u}[k] = [\boldsymbol{f}_1[k], \boldsymbol{f}_2[k], \boldsymbol{f}_3[k], \boldsymbol{f}_4[k], \boldsymbol{p}_1[k], \boldsymbol{p}_2[k], \boldsymbol{p}_3[k], \boldsymbol{p}_4[k]]$, as represented by Fig. 3b.

Given the open parameters $\boldsymbol{\xi}[k]$, the complete trajectory optimization problem can be formulated as follows:

$$\min_{\boldsymbol{\xi}} \sum_{k=0}^{N} l(\boldsymbol{\xi}[k]) + \phi(\boldsymbol{\xi}[N]) \qquad \text{(cost function)}$$

$$\text{s.t.} \quad \boldsymbol{x}[k+1] = f(\boldsymbol{x}[k], \boldsymbol{u}[k]) \qquad \text{(LL-SRBM dynamics)}$$

$$\boldsymbol{x}[0] = \boldsymbol{x}_{ini} \qquad \text{(initial states)}$$

$$\boldsymbol{x}[N] = \boldsymbol{x}_{fin} \qquad \text{(final states)}$$

$$t_{min} \leq dt[k] \leq t_{max} \qquad \text{(time limits)}$$

$$\|\boldsymbol{q}_B[k]\| = 1 \qquad \text{(quaternion norm)}$$

$$l_{min} \leq \|\boldsymbol{r}[k] - \boldsymbol{p}_i[k]\| \leq l_{max} \quad (i \in \{1, ..., N_l\}) \qquad \text{(kinematics limits)}$$

if foot in contact:

$$0 \leq f_i^z \leq f_{max}^z \quad (i \in \{1, ..., N_l\}) \qquad \text{(ground force limit)}$$

$$\boldsymbol{P}\boldsymbol{f}_i \leq \boldsymbol{0} \quad (i \in \{1, ..., N_l\}) \qquad \text{(friction cone)}$$

$$\boldsymbol{p}_i[k] = \boldsymbol{p}_i^* \quad (i \in \{1, ..., N_l\}) \qquad \text{(given contacts)}$$

if foot in the air:

$$\boldsymbol{f}_i = \boldsymbol{0} \quad (i \in \{1, ..., N_l\}) \qquad \text{(no contact force)}$$

B. Cost Function

The items inside the cost function can be categorized into 4 groups: *control inputs smoothness, energy consumption, time penalization* and *final state target*.

Control Inputs Smoothness: To generate a smooth motion, the difference of ground reaction forces and foot movements between adjacent nodes are minimized:

$$\sum_{k=0}^{N} \sum_{i=0}^{N_l} \left(\dot{\boldsymbol{p}}_i^2[k] + \dot{\boldsymbol{f}}_i^2[k]\right) \tag{11}$$

Energy Consumption: We minimize squared momentum to achieve minimal energy consumption:

$$\sum_{k=0}^{N} (\boldsymbol{H}^2[k] + \boldsymbol{L}^2[k]) \tag{12}$$

Time Penalization: We want the optimization to generate a minimum-time trajectory to finish a task while respecting the constraints. Also, this cost term encourages the robot to change the shape of the centroidal inertia in the *flight phase* to achieve a fast motion:

$$\sum_{i=0}^{N} (dt^2[i]) \tag{13}$$

Final State Target: Though final state targets are formulated as constraints, we find that putting final orientation target into cost function can speed up the solving process. This is especially the case in tasks requiring large change of orientation, like twisting jump. So we put the final orientation targets into the cost function:

$$(\boldsymbol{q}_B[N] - \boldsymbol{q}_B^{fin})^2 + (\dot{\boldsymbol{q}}_B[N] - \dot{\boldsymbol{q}}_B^{fin})^2 \tag{14}$$

C. Kinematics Constraint

The kinematics constraint bounds the distance between CoM and the feet to be within $[l_{min}, l_{max}]$,

$$l_{min} \leq \|\boldsymbol{r}[k] - \boldsymbol{p}_i[k]\| \leq l_{max} \quad (i \in \{1, ..., N_l\}) \tag{15}$$

For the bipedal robot SLIDER, $l_{min} = 0.35$ m and $l_{max} = 0.75$ m. For the quadrupedal robot ANYmal, $l_{min} = 0.31$ m and $l_{max} = 0.6$ m.

D. Contact Constraint

We have pre-defined the contact sequence as *takeoff phase*, *flight phase* and *landing phase*. We pre-define the start and target position of the foot, namely:

$$\boldsymbol{p}_i[k] = \boldsymbol{p}_i^{ini} \quad (i \in \{1, ..., N_l\}) \qquad \text{(takeoff phase)}$$

$$\boldsymbol{p}_i[k] = \boldsymbol{p}_i^{fin} \quad (i \in \{1, ..., N_l\}) \qquad \text{(post-landing phase)}$$

E. Ground Reaction Force Constraint

In [8] the actuator limits are introduced in the planner by approximating the configuration-dependent reaction forces. Although joint angles of the robot are not included in LL-SRBM, we can still approximate the ground reaction force limit using the default configuration by:

$$\boldsymbol{f}_{max} = \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}_0)\boldsymbol{\tau}_{max} \tag{16}$$

where $\boldsymbol{J}$ is the jacobian matrix, $\boldsymbol{q}_0$ represents the joint coordinates at the default configuration, $\boldsymbol{\tau}_{max}$ is the maximum joint torque vector. In *takeoff phase* and *landing phase* both feet need to be in contact with the ground, so all the ground reaction forces would be 0 or pushing against the ground, therefore the ground force limit is:

$$0 \leq f_i^z \leq f_{max}^z \quad (i \in \{1, ..., N_l\}) \tag{17}$$

We also ensure no slippage of foot contact points. The tangential forces are constrained to remain inside the Coulomb friction cone defined by the friction coefficient μ. We approximate the friction cone by the friction pyramid, which is a common approach to make the constraints linear and speed up the computation. The friction cone constraint is given by:

$$-\mu f_i^z \leq f_i^x \leq \mu f_i^z \tag{18}$$

$$-\mu f_i^z \leq f_i^y \leq \mu f_i^z \tag{19}$$

where f_i^x, f_i^y, f_i^z are components of the ground reaction force $\boldsymbol{f}_i$ $(i \in \{1, ..., N_l\})$.

If all feet is in the air, which is the case of *flight phase*, we enforce all ground reaction force to be exactly 0:

$$\boldsymbol{f}_i = \mathbf{0} \quad (i \in \{1, ..., N_l\}) \tag{20}$$

V. WHOLE-BODY CONTROL AND CONTACT DETECTION

The whole-body controller takes responsibility of computing the joint torques to achieve the desired motions while respecting a set of constraints. In the paper, the tasks of interest are the CoM position, the pelvis orientation, the angular momentum of the robot, the foot positions and orientations. Each task is comprised of a desired acceleration as a feed-forward term and a state feedback term to stabilize the trajectory. Generally, the task for the linear motion can be expressed as:

$$\boldsymbol{J}_{\mathrm{T}}\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}}^{\mathrm{cmd}} - \dot{\boldsymbol{J}}_{\mathrm{T}}\dot{\boldsymbol{q}},$$

$$\ddot{\boldsymbol{x}}^{\mathrm{cmd}} = \ddot{\boldsymbol{x}}^{\mathrm{des}} + \boldsymbol{K}_{\mathrm{P}}^{\mathrm{pos}}(\boldsymbol{x}^{\mathrm{des}} - \boldsymbol{x}) + \boldsymbol{K}_{\mathrm{D}}^{\mathrm{pos}}(\dot{\boldsymbol{x}}^{\mathrm{des}} - \dot{\boldsymbol{x}}),$$

where $\boldsymbol{J}_{\mathrm{T}}$ is the translational Jacobian for the task, $\boldsymbol{x}$ is the actual position of the link, and the superscript des indicates the desired motion.

For the task of angular motion, the command can be formulated as:

$$\boldsymbol{J}_{\mathrm{R}}\ddot{\boldsymbol{q}} = \dot{\boldsymbol{\omega}}^{\mathrm{cmd}} - \dot{\boldsymbol{J}}_{\mathrm{R}}\dot{\boldsymbol{q}},$$

$$\dot{\boldsymbol{\omega}}^{\mathrm{cmd}} = \dot{\boldsymbol{\omega}}^{\mathrm{des}} + \boldsymbol{K}_{\mathrm{P}}^{\mathrm{ang}}(\mathrm{AngleAxis}(\boldsymbol{R}^{\mathrm{des}}\boldsymbol{R}^{\mathrm{T}})) + \boldsymbol{K}_{\mathrm{D}}^{\mathrm{ang}}(\boldsymbol{\omega}^{\mathrm{des}} - \boldsymbol{\omega}),$$

where $\boldsymbol{J}_{\mathrm{R}}$ is the rotational Jacobian for the task, $\boldsymbol{R}$ and $\boldsymbol{R}^{\mathrm{des}}$ denote the actual and desired orientation of the pelvis link respectively, $\mathrm{AngleAxis}()$ maps a rotation matrix to the corresponding axis-angle representation, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the link.

For the CoM task and angular momentum task, the centroidal momentum matrix [16] is used as the task jacobian. For balancing or walking, angular momentum task are often defined as a damping task that damps out excess angular momentum. However for highly dynamic motion such as twisting jump, angular momentum varies a lot during the process and it plays a vital role to achieve the jump motion. In this case, the reference angular momentum is needed and it comes from our motion planner. Since the single rigid body model is used in the motion planner, its orientation and associated angular momentum are both well defined.

A. QP formulation

Inspired by [17], the full dynamics can be decomposed into the underactuated part and actuated part:

$$\begin{bmatrix}\boldsymbol{M}_f \\ \boldsymbol{M}_a\end{bmatrix}\ddot{\boldsymbol{q}} + \begin{bmatrix}\boldsymbol{H}_f \\ \boldsymbol{H}_a\end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ \boldsymbol{S}_a\end{bmatrix}\boldsymbol{\tau} + \begin{bmatrix}\boldsymbol{J}_f^{\mathrm{T}} \\ \boldsymbol{J}_a^{\mathrm{T}}\end{bmatrix}\boldsymbol{f},$$

where $\boldsymbol{M}$, $\boldsymbol{H}$, $\boldsymbol{S}_a$, $\boldsymbol{\tau}$, $\boldsymbol{J}$ and $\boldsymbol{f}$ are the mass matrix, Coriolis force matrix and gravitation force vector, the actuator selection matrix, joint torques vector, the stacked contact Jocabian and reaction force vector. The subscript, f and a, indicates the floating part and actuated part respectively. The weighted sum formulation is applied, in which one QP problem is solved at each control loop. The formulation of the QP problem can be written as

$$\begin{aligned}
\min_{\ddot{\boldsymbol{q}}, \boldsymbol{f}} \quad & \frac{1}{2}\|\boldsymbol{A}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}\dot{\boldsymbol{q}} - \boldsymbol{B}^{\mathrm{cmd}}\|_{\boldsymbol{W}_1}^2 + \frac{1}{2}\|\boldsymbol{f} - \boldsymbol{f}_{\mathrm{des}}\|_{\boldsymbol{W}_2}^2 \quad (21)\\
\text{s.t.} \quad & \boldsymbol{M}_f\ddot{\boldsymbol{q}} - \boldsymbol{J}_f^{\mathrm{T}}\boldsymbol{f} = -\boldsymbol{H}_f \qquad \text{(floating base dynamics)}\\
& \boldsymbol{P}\boldsymbol{f} \le \mathbf{0} \qquad \text{(friction cone)}\\
& \boldsymbol{S}_a^{-1}(\boldsymbol{M}_a\ddot{\boldsymbol{q}} + \boldsymbol{H}_a - \boldsymbol{J}_a^{\mathrm{T}}\boldsymbol{f}) \in [\boldsymbol{\tau}_{min},\, \boldsymbol{\tau}_{max}]\\
& \qquad \text{(input limits)}
\end{aligned}$$

where $\boldsymbol{A}$ is a stack of the Jacobian matrices for the tasks of interest, $\boldsymbol{B}^{\mathrm{cmd}}$ is a stack of the commanded accelerations and $\boldsymbol{W}_i\,(i = 1, 2)$ are the weighting matrices, $\boldsymbol{P}$ denotes the linearized friction cone matrix. Similar to [18][19] , the unilateral contact constraint is treated as a soft constraint by simply assigning a large weight on the desired zero acceleration. It is reported in [20] that this gives a better stability.

The output torque commands $\boldsymbol{\tau}$ at each control iteration is computed by

$$\boldsymbol{\tau} = \boldsymbol{S}_a^{-1}(\boldsymbol{M}_a\ddot{\boldsymbol{q}} + \boldsymbol{H}_a - \boldsymbol{J}_a^{\mathrm{T}}\mathbf{f}) \tag{22}$$

Since the SRBM is an approximate model, we set $\boldsymbol{W}_2 = \mathbf{0}$ to track the desired trajectory in the jump motion, except the landing phase. In the landing phase, in order to achieve soft landing, we switch to tracking desired contact forces by increasing $\boldsymbol{W}_2$ while decreasing the weights for centroidal momentum control to be $\mathbf{0}$. After a short safe landing phase (around 0.2 second), we switch back to trajectory tracking control in order to achieve the desired steady pose. This strategy of switching to force control during landing is similar to [8], which is particular critical for real robot experiments in the existing of trajectory tracking and state estimation errors.

B. Contact detection

According to [16], the spatial average velocity of a robot is defined as

$$\mathbf{v}_G = \begin{bmatrix}\boldsymbol{v}_{\mathrm{com}} \\ \boldsymbol{\omega}_G\end{bmatrix} = \boldsymbol{I}_f^{-1}\boldsymbol{h}_G$$

where $\boldsymbol{h}_G \in \mathbb{R}^6$ is the centroidal momentum of the robot, $\boldsymbol{I}_f \in \mathbb{R}^{6\times6}$ is the centroidal inertia matrix of the robot, $\boldsymbol{v}_{\mathrm{com}}$ is the center of mass velocity, $\boldsymbol{\omega}_G$ denotes the average angular velocity of the robot. Please refer to [16] for the details of $\boldsymbol{h}_G$ and $\boldsymbol{I}_f$.

We use the change rate of $\|\mathbf{v}_G\|_2$ to judge the contact. During the flight phase, the gravity is the only external force that changes the state of the robot. The norm of $\boldsymbol{v}_{\mathrm{com}}$ will increase under the gravity while the robot is dropping down from the peak. When contacts happen, the velocities will decrease. In order to respond to all kinds of contact, we choose to use $\|\mathbf{v}_G\|_2$ to detect contacts which is a comprehensive metric including all the changes of velocities of each link of a robot. When the numerical derivatives of $\|\mathbf{v}_G\|_2$ in a sampling window are all less than a threshold, we tell the controller that the robot gets contact. In practice, the proposed method is more reliable than using force signals that are very noisy.

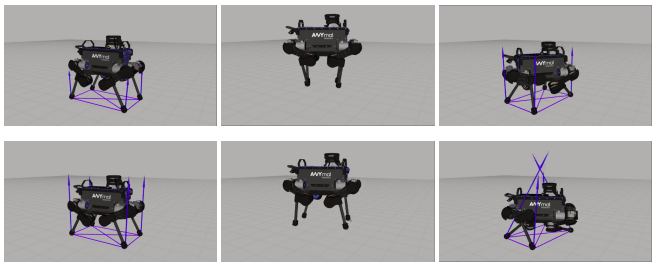


Fig. 5: Snapshots of the ANYmal robot performing a 90° twisting jump with inertia shaping and without inertia shaping. First row: twisting jump with inertia shaping. Second row: twisting jump without inertia shaping.

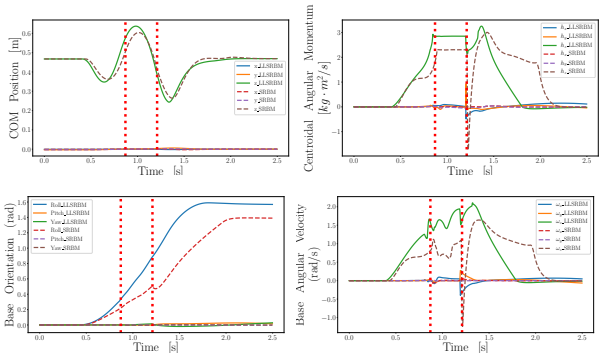


Fig. 6: The measured trajectories of ANYmal 90° twisting jump in simulation, with measured trajectories generated by LL-SRBM and SRBM repectively. In each plot the vertical dashed lines split the entire trajectory into three phases: *takeoff*, *flight* and *landing*. The full lines show the measured trajectories with LL-SRBM and the dashed lines show measured trajectories with SRBM.

VI. EXPERIMENT RESULTS

A. Robot Platforms

SLIDER is a knee-less bipedal robot designed by the Robot Intelligence Lab at Imperial College London [21], [22]. It is 1.2 m tall and has 10 actuated joints with a total weight of 16 Kg. Most of its weight is concentrated in the pelvis. The prismatic knee joint design is an unique feature of this robot that differentiates it from many other robots with anthropomorphic design. Also the sliding joint has a relatively large range of motion. The overall light weight and large range of leg motion make the robot suitable for agile locomotion.

The quadrupedal robot ANYmal is made by ANYbotics. It has 12 SEAs (Series Elastic Actuators) and weighs approximately 35 Kg. When ANYmal standing still in the default configuration, it is about 0.5 m tall.

B. Implementation

The trajectory optimization framework is implemented in CasADi [23] with Python using the interior-point solver IPOPT [24]. We write the whole body controller with C++ using the Pinocchio library [25] to compute full rigid body dynamics and qpOASES [26] to solve the QP problem. For SLIDER robot the whole body controller runs at 1k Hz to track the desired trajectory. For ANYmal the whole-body controller is running at 400 Hz in simulation and real experiments as well. For both robots the simulation is done in Gazebo with ODE as the physics engine.

C. Twisting Jump

We experimented the twisting jump on both ANYmal and SLIDER robots and compared the performance between the trajectories generated by LL-SRBM and SRBM. In the trajectory optimization we simply provide a linear interpolation of the orientation as the initial guess. Also we assign a large weight to the *final state target* and *time penalization* in the cost function to encourage the solver to find a trajectory that reaches the target pose in a minimum time.

As shown in Fig. 5, the foot motion generated with LL-SRBM has a big difference compared to the one generated with SRBM. In the flight phase, the feet try to keep close to the central rotating axis with the LL-SRBM generated trajectories. The SRBM generated foot trajectories just do an interpolation from the start to the goal position. This interesting behaviour emerges as LL-SRBM takes leg dynamics into consideration in the planning and tries to maximize the angular velocity in flight phase by modifying the shape of inertia with the leg motion. Figure 6 shows the measured trajectories of the twisting jump experiments. It can be seen clearly that in flight phase the base angular velocity in experiments with LL-SRBM does not drop as much as the base angular velocity in experiments with SRBM. The robot reaches the goal orientation earlier with LL-SRBM. To accomplish the twisting jump for SLIDER, we have added two joints for the robot, one yaw joint per leg. SLIDER can perform a 90° twisting jump and the motion can be seen in the accompany video.

In real experiments, we demonstrate the effectiveness of the proposed approaches by twisting jumps on unknown objects, as shown in Fig 8. A wooden board with a height of 25 mm was put under ANYmal when it was in the flight phase. Figure 7 shows the norm of the average spatial velocity. Even though collected from real robot experiments, the norm of the average spatial velocity is smooth and can be used as a reliable judgement for contact detection. The robot successfully detected the contact event and switched

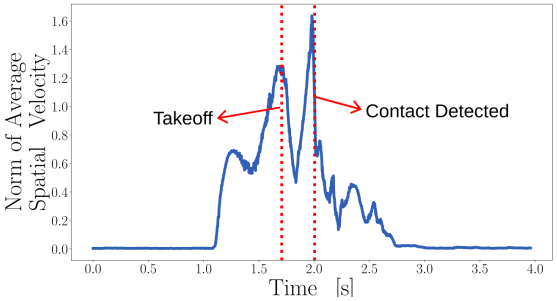


Fig. 7: The norm of average spatial velocity when ANYmal performed a twisting jump. A wooden board was put under the robot while it was flying. A big drop of the value after getting contact when the robot was dropping down.

(a) (b) (c) (d)

Fig. 8: Snapshots of ANYmal performing a twisting jump with 30°. We also put a wooden board with a height of 25 mm under the robot when it was in the *flight* phase to test the contact detection and the robustness of our controller. From left to right: (a): takeoff phase, (b): flight phase, (c): contact is detected. (d): the robot is stable after a soft landing.

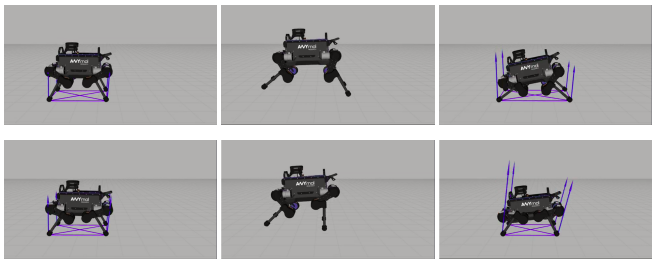


Fig. 9: Snapshots of the ANYmal robot performing a forward jump of 30 cm with inertia shaping and without inertia shaping. First row: forward jump with inertia shaping. Second row: forward jump without inertia shaping.

to a soft landing.

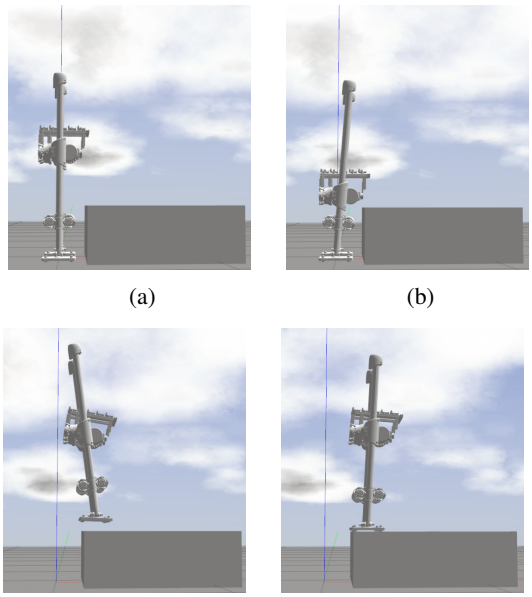
D. Forward Jump

In the trajectory optimization, we assign a large weight to the *final state target* and *energy consumption*. For ANYmal, a forward jump motion of 30 cm is generated with LL-SRBM and SRBM, as show in Fig. 9. It can be seen that for trajectories generated with LL-SRBM model, the feet stretch out more in the air than trajectories generated with SRBM. This is because the optimizer tries to increase the inertia around y axis to reduce the change of base pitch angle. The stretched feet in the flight phase also help to prevent early touchdown of the robot.

We also tried jump onto a box for SLIDER with LL-SRBM. The maximum height of the box SLIDER can jump on is 35 cm, with a forward jump length of 20 cm, as shown in Fig. 10. Considering that SLIDER is 1.2 m high, the robot can jump onto a box equal to 30% of its total height. Compared with the anthropomorphic robot design which has knees, SLIDER's straight legs have a larger range of motion, allowing SLIDER to reach the same jump height with a relatively smaller CoM height. Because there are no knee joints on legs of SLIDER, the feet can go all the way up until the ankles touch the pelvis. That is the reason why SLIDER can jump onto a high box.

E. Computation Time

We recorded the computation times of trajectory optimizations for various dynamic jump motions. For the bipedal robot we used 60 knots and each knot has 59 variables (26 for states and 33 for control inputs). For the quadrupedal robot we used 50 knots and each knots has 53 variables (26 for states and 27 for control inputs). Although the complexity



(a) (b) (c) (d)

Fig. 10: Snapshots of SLIDER performing a forward jump onto a 35 cm box using LL-SRBM. (a) ∼ (b): takeoff phase, (c): flight phase, (d): landing phase.

of LL-SRBM has increased compared with SRBM, the computation time with LL-SRBM only increase by an average of 33.6% with respect to that with SRBM, as shown in Table I.

TABLE I: Solve Times for Planners.

	quadruped forward jump	**quadruped twist jump**	**biped forward jump**	**biped twist jump**
SRBM	1.50s	2.45s	6.33s	5.44s
LL-SRBM	1.53s	2.56s	9.54s	7.37s

VII. CONCLUSIONS

This paper proposes an unified model with inertia shaping for planning highly dynamic jumps of legged robots. This model allows the motion planner to improve the jumping performance by actively changing the centroidal inertia. In the meanwhile, this paper also proposes a novel contact detection method using the norm of average spatial velocity. The twisting jump and forward jump experiments on bipedal robot SLIDER and quadrupedal robot ANYmal show the improved jump performance after using the proposed model

and the robustness of the controller to unforeseen impacts. In the future, we are interested in speeding up the computation to re-plan the jump motion with the proposed LL-SRBM.

APPENDIX

QUATERNION TO ROTATION MATRIX CONVERSION

Given a quaternion $\boldsymbol{q} = q_w + q_x\boldsymbol{i} + q_y\boldsymbol{j} + q_z\boldsymbol{k}$ which represents the orientation of the single rigid body in world frame, the equivalent rotation matrix representation can be derived as:

$$\boldsymbol{R} = \begin{bmatrix} 1-2\left(q_y^2+q_z^2\right) & 2q_xq_y - 2q_wq_z & 2q_wq_y + 2q_xq_z \\ 2q_xq_y + 2q_wq_z & 1-2\left(q_x^2+q_z^2\right) & 2q_yq_z - 2q_wq_x \\ 2q_xq_z - 2q_wq_y & 2q_wq_x + 2q_yq_z & 1-2\left(q_x^2+q_y^2\right) \end{bmatrix}$$

DERIVATIVE OF QUATERNION

Followed by [27], given a rigid body with quaternion $\boldsymbol{q} = q_w + q_x\boldsymbol{i} + q_y\boldsymbol{j} + q_z\boldsymbol{k}$ and with angular velocity $\boldsymbol{\omega}$, the derivative of the quaternion $\dot{\boldsymbol{q}}$ can be calculated as:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \circ \boldsymbol{\omega}$$

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{q}_z \\ \dot{q}_w \end{bmatrix} = \frac{1}{2}\begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} \circ \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{q}_z \\ \dot{q}_w \end{bmatrix} = \frac{1}{2}\begin{bmatrix} q_w & -q_z & q_y & q_x \\ q_z & q_w & -q_x & q_y \\ -q_y & q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \\ 0 \end{bmatrix}$$

ACKNOWLEDGMENT

This work is supported by the CSC Imperial Scholarship, EPSRC UK RAI Hubs NCNR (EP/R02572X/1), FAIR-SPACE(EP/R026092/1). The authors would like to thank Digby Chappell for helpful discussions.

REFERENCES

[1] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[2] I. Mordatch, M. De Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," in *ACM SIGGRAPH 2010 papers*, 2010, pp. 1–8.

[3] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5134–5140.

[4] ——, "Development of high-span running long jumps for humanoids," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 222–227.

[5] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.

[6] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[7] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[8] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors," 2021.

[9] T. Corbères, T. Flayols, P.-A. Léziart, R. Budhiraja, and N. Mansard, "Comparison of predictive controllers for locomotion and balance recovery of quadruped robots," in *2021 IEEE International Conference on Robotics and Automation - ICRA*, Xi'an, China, May 2021. [Online]. Available: https://hal.laas.fr/hal-03034022

[10] M. Halm and M. Posa, "Modeling and analysis of non-unique behaviors in multiple frictional impacts," *arXiv preprint arXiv:1902.01462*, 2019.

[11] T. Stouraitis, L. Yan, J. Moura, M. Gienger, and S. Vijayakumar, "Multi-mode trajectory optimization for impact-aware manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9425–9432.

[12] W. Yang and M. Posa, "Impact invariant control with applications to bipedal locomotion," 2021.

[13] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[14] S.-H. Lee and A. Goswami, "Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 4667–4672.

[15] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[16] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2, pp. 161–176, 2013.

[17] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, p. 473–491, Aug 2015. [Online]. Available: http://dx.doi.org/10.1007/s10514-015-9476-6

[18] T. Apgar, P. Clary, K. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot cassie." in *Robotics: Science and Systems*, vol. 101, 2018, p. 14.

[19] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, no. 3, pp. 429–455, 2016.

[20] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 120–127.

[21] K. Wang, D. Marsh, R. P. Saputra, D. Chappell, Z. Jiang, A. Raut, B. Kon, and P. Kormushev, "Design and control of slider: An ultra-lightweight, knee-less, low-cost bipedal walking robot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 3488–3495.

[22] K. Wang, A. Shah, and P. Kormushev, "Slider: A bipedal robot with knee-less legs and vertical hip sliding motion," in *Proc. 21st International Conference on Climbing and Walking Robots and Support Technologies for Mobile Machines (CLAWAR 2018)*, Panama, September 2018.

[23] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[25] J. Carpentier, F. Valenza, N. Mansard *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2021.

[26] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[27] B. Graf, "Quaternions and dynamics," *arXiv preprint arXiv:0811.2889*, 2008.